

The Denoising Convolutional Autoencoder

Luca Rinaldi

luca.rinaldi.3@studenti.unipd.it

Marco Giunta

marco.giunta.1@studenti.unipd.it

Abstract

We analyzed the application of the Convolutional Autoencoder architecture to the task of denoising images of highly obfuscated human faces images. In particular we trained several variations of this architecture to accept as input images from the Labeled Faces in the Wild, which we corrupted via Gaussian blur and pixelation. We compare performance across different architectures using a variety of different metrics (PSNR, SSIM, human aesthetics perception). Finally we explore the role of the loss function by training with some loss functions common in the literature (pixel loss, perceptual loss, and a weighted superposition of the two). In the end we obtain that all variations of the CAE architecture are able to effectively reconstruct highly obfuscated faces with a quick training consisting of only a few epochs, which can be further reduced by using the perceptual/mixed loss.

1. Introduction

When divulging sensitive photographic media it's common practice to artificially corrupt images to protect the privacy of the depicted subjects. To avoid privacy leaks recognizable details such as facial features are tampered, e.g. via pixelation (a.k.a. mosaicing), blurring, or more sophisticated techniques like P3, which encrypts some coefficients in the JPEG representation of the image. The need for privacy-preserving tools goes beyond its historical applications, like the publishing of police reports, and has exploded with the rise of user-contributed photos and videos on online platforms; for example YouTube offers a tool to automatically detect and blur faces of passerbying people in the background of videos captured in public spaces. However several studies show that images corrupted using these common obfuscation techniques can still retain enough useful information that the originals may be recovered; therefore it's interesting to test whether the bar for the design of privacy-preserving technologies should be raised - which can be done e.g. by showing that an appropriate deep-learning based approach is able to effectively reconstruct the uncorrupted image.

It's easy to think other situations where such technologies may be useful: for example as a way to restore blurred pictures (e.g. out of focus photos), or to counteract the loss of information caused by the compression and/or the noise added by video streaming services.

The purpose of this work is to use some variations of the *Convolutional Autoencoder* neural network architecture to reconstruct images of faces corrupted with gaussian blur/pixelation; in particular we test what can be achieved with these architectures and limited computational resources, whether the capabilities of the CAE can be improved by replacing/combining the standard pixel loss with a more sophisticated perceptual loss, analyze performance with common metrics (PSNR, SSIM, human aesthetics perception), and finally comment on the results by measuring the quality of the outputs against the resources needed to obtain them.

Notice that this denoising task, while related to the super-resolution task (they both aim to improve low-quality images), is distinct: in the former the output image has the same dimensions as the input (with hopefully only the noise removed), whereas in the latter the output is larger than the input.

2. Related Work

Convolutional neural networks (CNN) are a deep neural network architecture commonly used for object recognition and image classification, known to yield high prediction accuracy in supervised learning tasks; they “*provide a way to specialize neural networks to work with data that has a clear grid-structured topology and to scale such models to very large size. This approach has been the most successful on a two-dimensional image topology*”. [2] Their effectiveness when dealing with images stems from their ability to capture multiple local features (via the multi-channel feature maps) while preserving the original spatial structure of the image. This makes them useful in many facial recognition tasks; for example CNNs have been proved to be able to reach near 100% accuracy level in the classification task applied to the LFW dataset [17] [12]. CNNs are known to be very accurate in classifying faces even if they have been previously blurred or pixelated [11]; this makes

them the natural candidate for building a simple but effective denoiser. In order to actually build our network we turn to the *autoencoder* generative architecture, whose applications range from dimensionality reduction/feature learning to facial recognition [2]. The idea behind a denoising autoencoder is to perform “smart compression”: we map the input to a lower dimensional space, thus forcing some information to be discarded; if learning is successful only the irrelevant noise will be thrown away, resulting in a “cleansed” image [2]. In practice this means sending the images through an *encoder* made of several layers (which we choose to be convolutional for the above reasons); then the smaller dimensional representation goes through a *decoder*, which is a specular copy of the encoder. The result will be an image with the same resolution as the input. This *convolutional autoencoder* (CAE) architecture is known to work well in facial recognition classification tasks [3] as well as denoising ones [15]. For these reasons we based our CAE architecture on [15], where the authors also aim to denoise blurred/pixelated images from the LFW dataset; in addition to a base CAE architecture we also develop three improved versions:

- The *variational convolutional autoencoder* (VCAE), where we inject extra stochasticity at the latent space level. This has been shown to sometimes be able to boost performance in the denoising task [6].
- The CAE with *residual blocks* (ResBlocks), which can also improve model performance [8].
- The base CAE but trained with one of two versions of the *Perceptual Loss* based on VGG16, which has been shown to be able to significantly improve results both in a denoising setting [15] as well as in a super-resolution one [7].

To quantify model performance we use both the standard *peak signal to noise ratio* (PSNR) as well as the more sophisticated *structural similarity index measure* (SSIM) [16]. At any case we remark that the final performance index must be human aesthetics, as these metrics are known to not correlate well with human perception of result quality; several studies show that that PSNR in particular is often outperformed by other visual quality metrics [10] [14] [1] [5], and we expect this to be the case in our problem too [15].

3. Dataset

The *Labeled Faces in the Wild* dataset [4] was used to train all architectures. This dataset consists of 13,233 250×250 JPEG images collected from the web, depicting the faces of celebrities. We used the *funneled version* of the dataset, which adds some preprocessing to the images; in particular faces are detected and centered using the

openCV implementation of the Viola-Jones face detector, then cropped and enlarged by a factor of 2.2 [4], resulting in 224×224 JPEG images. These measures ensure heads are fully captured in the image and at an almost uniform size.

We then added some further preprocessing by performing a center crop, so that the images are now 112×112 ; this is done to ignore most of the backgrounds, since we want to focus on the facial region for the obfuscation task. Finally before passing an image to the network we normalize it by mapping it to the $[0, 1]$ interval, then adjusting the mean and the std such that they both equal 0.5; this is done to improve the performance of the network, since empirically we noticed it helped accelerate convergence.

To train the networks we obfuscated the images mostly via *gaussian blur*, which consists in convolving the 2D matrix specified by the image with a gaussian kernel whose size and variance parameter σ must be fixed by the user; this results in an image where finer details are removed. For our experiments we used a gaussian kernel with size 9 and $\sigma = 4$.

The second obfuscation method is pixelation a.k.a. mosaicing, which consists in dividing the image into $n \times n$ squares, which are then painted with the average color of the pixels in that square. We used $n = 8$ to stress test the capabilities of the CAE architecture, as this is a more aggressive way of obfuscating the image.

The values $\sigma = 4$ and $n = 8$ were chosen as they represent a good compromise: the resulting obfuscated images are corrupted enough that facial features appear to be lost to the human observer, but not to the point that a simple CAE with reasonably limited training resources wouldn’t be able to achieve good enough performance.

Finally we report that of the 13,233 images about 30% were randomly picked to be the test set; 20% of the remaining ones were randomly picked to be the validation set, with the remaining being the training set (training: 7620 samples, validation: 1905 samples, test: 3708 samples).

4. Methods

4.1. Architecture

4.1.1 Base architecture

The base model we used to reconstruct obfuscated human faces is a convolutional autoencoder with an input and output shape of $122 \times 112 \times 3$. The encoder consist of 4 convolutional layers, in which the first one has $64 \ 9 \times 9$ filters. The following three layers consist of 64, 128 and $256 \ 4 \times 4$ filters respectively with stride 2. The decoder can be seen as the inverse of the encoder: its first three layers are transpose convolutional layers with 256, 128 and $64 \ 4 \times 4$ filters respectively with stride 2. The last one consists of $3 \ 9 \times 9$ filters. All layers are followed by batch normalization and

ReLU activation, except the last layer which is followed by a tanh activation. Such an architecture produces a latent tensor of shape $256 \times 12 \times 12$.

An optional visual representation of this architecture is the one corresponding to the green line in fig. 5.

4.1.2 Improved architectures

Variational CAE The Variational CAE (blue line in fig. 5) encodes instead a distribution which is regularized during the training in order to ensure that its latent space has good statistical properties regarding the generative process - namely: continuity and completeness [2] [9] [13]. The decoder is fed with a sample from the encoded distribution. We chose the encoded distributions to be normal so that the encoder can be trained to return the mean and the covariance matrix that describes these Gaussians, as it is customary with this type of architecture [9]. We implemented those computations feeding in parallel the latent vector produced by the standard encoder to 2 convolutional layers with $256 \ 1 \times 1$ filters followed by batch normalization, producing the mean and covariance tensors, which are now combined with the VAE reparametrisation trick [9], making gradient descent possible despite the random sampling. The loss function now consists of an additional term that regularises the encoded distribution to be close to a standard Gaussian - which, once again, is customary with this type of architecture [9].

ResBlocks We also tested a deeper architecture (ResCAE), in which we appended to the encoder 3 residual blocks (red line in fig. 5). A block consist of two convolutional layers with $256 \ 3 \times 3$ filters, each of one followed by batch normalization and ReLU activation; to each of their output is summed the output of the previous block.

4.2. Loss Function

4.2.1 Pixel loss

The first type of loss we implemented is the *pixel loss*, which is the MSE (mean squared error, i.e. normalized Euclidean distance) loss between the original and the reconstructed image:

$$\ell_{\text{pixel}}(x, \hat{x}) = \frac{1}{112 \times 112 \times 3} \|x - \hat{x}\|_2^2 \quad (1)$$

where the $112 \times 112 \times 3$ is simply the number of pixels in our 112×112 RGB images.

The pixel/MSE loss is a common choice with generative models dealing with pictures. This makes sense because this loss is the normalized sum of the squared differences between corresponding pixels, and therefore it penalizes elementwise differences between the two images.

Still it has the issue of being “naive”: images that look similar to a human observer can be made to have relatively large pixel losses by having many small elementwise differences, and similarly relatively low losses can correspond to images that a human observer could clearly tell apart [15]. This can be improved by switching to a different loss, that can also significantly reduce the number of epochs needed to reach convergence.

4.2.2 Perceptual Loss

A loss that minimizes pixel differences captures low level features; a better way to approach the problem may be to look for one that minimizes differences between high level features. For example if the facial features are correctly reproduced with slightly different colors a human observer would say the result is good, even though the individual pixels may be significantly different. This can be achieved using transfer learning, to exploit previously gained knowledge about these high level features (e.g. the shape of the eyes, hair, etc.).

In this work we use the *perceptual loss* implemented in [7], defined as follows:

$$\ell_{\text{perceptual}}(x, \hat{x}) = \frac{1}{H \times W \times C} \|\varphi(x) - \varphi(\hat{x})\|_2^2 \quad (2)$$

The perceptual loss is the MSE computed between the feature representations $\varphi(x)$, $\varphi(\hat{x})$ of the two images, whose shapes are (H, W, C) . Following [7] we define φ as the output of the `relu2_2` layer of the VGG16 network pre-trained on ImageNet with its weights fixed. This is useful because the first few convolutional layers of this network contain information about high level features - whereas the latter, more specialized neurons are not used to define our loss network. Notice that VGG16’s expected input has a shape equal to $(224, 224, 3)$, so in order to implement our perceptual loss we added an upsampling layer with linear interpolation to double the length size; the combination of this upsampling layer, the first 15 VGG16 layers, and finally the MSE constitutes the loss network we used to better capture the differences in content features in the images.

Finally notice that this feature representation has shape $(H = 56, W = 56, C = 256)$.

4.2.3 Mixed Perceptual Loss

One may also wonder whether combining the two losses could allow us to capture both high and low level features, i.e. reconstruct images that look good to a human while also being accurate on the level of individual pixels. For this reason we tested a third *mixed perceptual loss* given by:

$$\ell_{\text{mixed}} = \rho \ell_{\text{pixel}} + \ell_{\text{perceptual}} \quad (3)$$

The ρ parameter quantifies the relative importance of each contribution: for example if $\rho = 1$ the two terms are weighed equally, whereas if $\rho = 0.5$ the perceptual loss is twice as important as the pixel one.

5. Experiments

We performed a small-scope grid search to choose the hyperparameters for the CAE that optimize the PSNR / SSIM scores; this allowed us to choose a batch size of 8, an Adam optimizer with an initial LR of $5 \cdot 10^{-4}$, and a small L2 regularization with a 10^{-5} strength. We also chose to run training for up to 15 epochs, as this was a good compromise between performance (the loss always flattened at around 15 epochs, and often stopped significantly improving after 5-10 epochs) and training time (using between 5 and 15 epochs we were able to keep each training session under about an hour with the available GPUs).

5.1. Architecture experiment

The first experiment was testing the base architecture, the CAE, evaluating its performances in reconstructing obfuscated human faces by the means of PSNR/SSIM. As can be seen from figures 1, 2, the results are already quite good, and we can appreciate the fact that the autoencoder was able to successfully retrieve the useful information and discard the irrelevant noise. These results will be the baseline for the comparison with the results of other models.

The second experiment consisted in the implementation of the variational architecture, where the noise is injected in the latent space. At first we tried to compute the tensors μ and σ , which fully characterize the latent distribution, by the means of linear layers, but unfortunately this was not feasible if we were to maintain the same dimension of the latent space, as it would have resulted in at least $(36864)^2$ parameters. In order to maintain the same latent space dimension we decided to generate μ and σ according to Subsection 4.1.2. Unfortunately the quality of the reconstruction is worse than the standard CAE, as can be seen in figures 1, 2 and 3. If we try to give less importance to the distribution's regularization term of the loss we notice that the results improve, and taking the limit of this procedure will ensure the results approach the ones of the base CAE. This is easily explained: with no regularization on latent space distributions, these could degenerate to a Dirac Delta i.e. distributions with no variance, hence neglecting the injected noise and behaving as a standard CAE [9] [2]. Our guess is that a convolutional layer with a 1×1 kernel is not able to effectively compute adequate μ and σ estimates for the latent tensor. In order to do so we would instead need a dense layer like a linear one, but conserving the latent space dimension would result in a number of parameters that exceeds our computational constraint.

Another experiment we performed was testing the enhanced architecture obtained by deepening the encoder with three ResBlock. We noticed that each training epoch was $\sim 30\%$ slower than CAE, and the results worsened by 7% and 5% for PSNR and SSIM respectively, even though according to human aesthetics there were no relevant worsening and sometimes the ResCAE reconstruction seemed to actually outperform the basic CAE.

5.2. Perceptual Loss experiment

We then retrained the basic CAE architecture, but with the VGG16-based *perceptual loss* described above. The main drawback of this loss is that it makes the computations more complex; in our case it made each training epoch about 2-3 times slower. This makes sense: now the signal has to travel through 15 extra VGG16 layers - and even though we don't need to compute extra gradients as the VGG weights are fixed the same gradients take longer to compute. And yet thanks to the slower epochs we obtain much better results, as can be observed in figures 1 and 2. Despite the lower PSNR / SSIM scores the images obtained with the perceptual loss-based architecture look quite a bit better: they're generally more aesthetically pleasing, and look closer to the originals. The outputs look smoother and retain more details from the base image, whereas the pixel loss produces images that look more blocky, grainy, and with less features. This is especially evident if we focus on the teeth (fig. 1): the pixel loss-based images struggle reproducing the fine lines between the teeth (which can be successfully reconstructed with the perc. loss.). Similarly the pixel loss-based architecture generally paint eyes which are black blobs, while the perceptual loss-based ones typically reconstruct correctly the white of the eyes and the distinction between pupil and iris (fig 1).

The other main benefit of using the perceptual loss is speeding up the *overall* training time. Even though each individual epoch takes more time we consistently observed that even after just 1 or 2 epochs the images obtained with the perceptual loss were already more detailed than what their pixel loss-based counterparts could output in 10+ epochs. This can be observed by noticing that the last three columns of figures 1, 2 were obtained with a 5 epochs training, whereas the others with a 15 epochs one. This makes the perceptual loss the clear winner in contexts where either the visual quality or the minimum number of training epochs to reach satisfaction must be optimized. Appendix figure 6 is a visual representation of this result.

The only area where the basic perceptual loss (2) struggled a bit was getting the colors right: even though the facial details were effectively reconstructed basically immediately the output images of early epochs were somewhat pale, with the colors getting less and less washed-out as the training went on; this is especially obvious in figure 2, where with


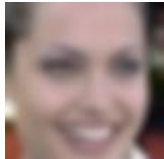


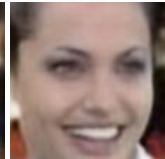
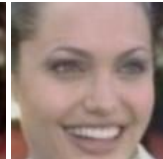
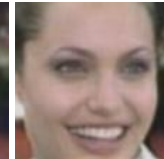
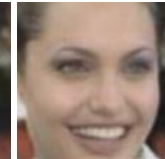
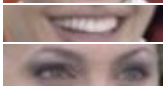
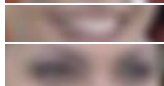

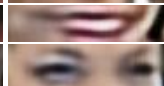
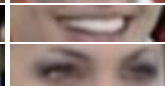
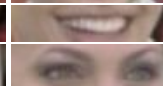

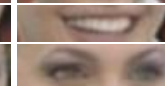
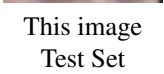
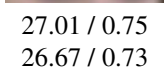
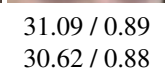
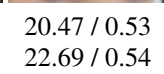
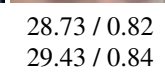
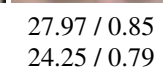
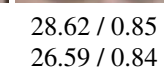
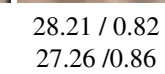
Ground truth	Blurred	CAE	VCAE	ResCAE	CAE + PL	CAE/MPL0.5	CAE/MPL 1
							
							
							
This image	27.01 / 0.75	31.09 / 0.89	20.47 / 0.53	28.73 / 0.82	27.97 / 0.85	28.62 / 0.85	28.21 / 0.82
Test Set	26.67 / 0.73	30.62 / 0.88	22.69 / 0.54	29.43 / 0.84	24.25 / 0.79	26.59 / 0.84	27.26 / 0.86

Figure 1: Output comparison of a sample dataset image across different tested architectures, with zoom on eyes and teeth; the PSNR / SSIM for the given example and for the dataset are reported. Notice that the main difference is determined by the loss: images obtained with any version of the perceptual loss look closer to the original and more detailed (cfr the white of the eyes and the distinction between teeth).

Ground truth	Blurred	CAE	VCAE	ResCAE	CAE + PL	CAE/MPL0.5	CAE/MPL 1
							
							
This image	27.45 / 0.78	31.62 / 0.91	21.84 / 0.58	28.73 / 0.82	25.46 / 0.83	27.90 / 0.88	29.01 / 0.91
Test Set	26.67 / 0.73	30.62 / 0.88	22.69 / 0.54	29.43 / 0.84	24.25 / 0.79	26.59 / 0.84	27.26 / 0.86

Figure 2: Output comparison of a sample dataset image across different tested architectures, with zoom on the eyes. The PSNR / SSIM for the given example and for the dataset are reported.

only 5 epochs the CAE + PL architecture is not yet able to make the picture dark enough. The opposite is true for the pixel loss: the colors are quickly recovered, whereas the high level facial features take quite a bit of time to be effectively reconstructed.

For this reason we tried combining the two losses using the *mixed perceptual loss* (3), the idea being that the mixed loss CAE architecture could accurately reproduce both the colors and the face details after a small number of epochs. In particular we used three versions of the perceptual loss: a “pure” one, a mixed one with equal weights, and one with a “half” pixel loss i.e. we tried $\rho = 0, 1, 0.5$ in eq. (3). The result is as expected, and can be observed in figure 2: after a 5 epochs training $\rho = 0$ is still too pale, $\rho = 0.5$ is a bit better but still not enough, and $\rho = 1$ comes closest to the original picture. This confirms that a strong enough contribution from the pixel loss can accelerate the learning of color reconstruction, so that the pixel loss can still offer a useful contribution to an otherwise clearly superior perceptual loss.

Of course even with the perceptual loss the CAE has some limitations - for example reproducing thin glasses, as can be seen in figure 3. Glasses are difficult for the model to gen-

erate because of their small edges in the ground truth which basically disappear in the blurred image. Indeed notice that in the pixel loss-based architectures the glasses’ legs are barely visible, while the lenses are not; this slightly improves with the perceptual loss, where the lenses reappear - but just barely. From this we conclude that even though these models have easy to find flaws these limitations are very reasonable, and therefore the CAE works quite well - especially with the perceptual loss, in good accord with the previous findings.

Finally notice that we were able to reproduce the many studies about PSNR / SSIM ([15] in particular): except for the VAE, whose training was unsuccessful, we observe that the simplest architecture has the best scores but the worst images according to human aesthetics, whereas the networks that produce the better looking images consistently score worse. This is in good accord with the idea that PSNR / SSIM don’t correlate well with human perception, and therefore are poor metrics compared to visual quality.

5.3. Pixelation experiment

To push the limits of the denoising CAE we tried repeating the training on a smaller scale with a more aggressive

Ground truth	Blurred	CAE	VCAE	ResCAE	CAE + PL	CAE/MPL0.5	CAE/MPL 1
							
							
This image	25.42 / 0.63	29.36 / 0.82	21.20 / 0.49	28.73 / 0.82	24.24 / 0.74	27.48 / 0.81	27.32 / 0.80
Test Set	26.67 / 0.73	30.62 / 0.88	22.69 / 0.54	29.43 / 0.84	24.25 / 0.79	26.59 / 0.84	27.26 / 0.86

Figure 3: Output comparison of a sample dataset image across different tested architectures, with zoom on the eyes. The PSNR / SSIM for the given example and for the dataset are reported.


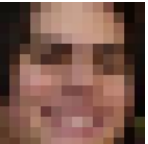
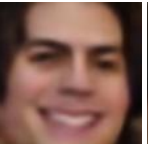

Base	Pixelated	CAE	CAE/MPL1
			
This image	21.20 / 0.44	25.83 / 0.75	22.42 / 0.68
Test Set	22.42 / 0.48	25.40 / 0.68	22.48 / 0.61

Figure 4: Output comparison of a sample (pixelated) dataset image fed to a CAE with either the pixel loss or the mixed equal-weights one. The PSNR / SSIM for the given example and for the dataset are reported.

corruption of the input images, via *pixelation* (*mosaicing*). The results can be seen in figure 4. We notice that compared to the blurred dataset here more details are removed, resulting in an image with less details and more blur. We also notice that due to the resolution of the pixelation the subject’s left eye and eyebrow merge in a single brown block, which the network interprets as a tilted “Picasso style” eye. This is actually somewhat correct: that single block is where an eye would usually be found, so it makes sense for the architecture to guess so. Still this result shows that this architecture is clearly limited when it comes to stronger image tampering methods, and not even the perceptual loss can save the network from inaccurately reconstructing low-resolution pixelated inputs (cfr 4): even with the more sophisticated loss the lack of definite, not blurred details and the “Picasso eye” cannot be removed - even though the eye tilt is not as extreme/noticeable in the last column of fig. 4, meaning the mixed perceptual loss can still offer somewhat of an improvement.

This problem did not appear before because blurring is able to reduce detail while still maintaining the relative position of various facial features (e.g. eyes, mouth, nose, etc.); with pixelation, instead, a lot of specific location details are lost within each pixelated block (e.g. the eye-eyebrow block previously discussed).

This seems to suggest that stronger architecture modifications are needed to defeat this noise injection technique.

6. Conclusions

The problem of denoising blurred / pixelated faces from the Labeled Faces in the Wild dataset has been tackled with several variations of the convolutional autoencoder architecture; the main results of this study are as follows.

- The CAE generative architecture is quite capable of reconstructing satisfactory approximations of the original image when it comes to gaussian blur image corruption, and can be considered relatively successful even in challenging situations (e.g. fig. 3); instead it somewhat struggles with strong pixelation, but reasonably so, as has been discussed.
- We found that simple modifications to the basic CAE architecture (e.g. adding variational/residual blocks) yield marginal performance improvements at best (at least with the ResCAE architecture; we were not able to successfully train a VAE to reach the same performance level). A significant boost in perceived reconstruction quality, as well as in min. number of epochs needed to reach satisfactory performance can be obtained by switching from the pixel loss to a perceptual one as described above. By combining the strengths of the two losses convergence can be further accelerated. Future studies may explore what happens by performing different combinations of architecture parameters, e.g. by using the perceptual loss with the variational / resblocks CAE.
- As discussed above it’s well established in the literature that PSNR / SSIM do not necessarily imply aesthetically pleasing reconstruction. We were able to reproduce this result, since in every example shown the best reconstructions also had the worst scores (except for the VAE results, which have bad scores since as discussed its training was unsuccessful). This poses a

problem to be tackled in future studies: how can one define a *quantitative* metric that still agrees with human intuition? This may be an interesting open problem.

We finally conclude by remarking that our study shows that (especially with limited computational resources) the denoising CAE can be an effective alternative to more expensive, state of the art solutions such as GAN-based architectures.

References

- [1] M. Ghanbari. “Scope of validity of PSNR in image/video quality assessment”. English. In: *Electronics Letters* 44 (13 June 2008), 800–801(1). ISSN: 0013-5194. URL: https://digital-library.theiet.org/content/journals/10.1049/el_20080522.
- [2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [3] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. “Transforming Auto-Encoders”. In: (2011). Ed. by Timo Honkela et al., pp. 44–51.
- [4] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007.
- [5] Q. Huynh-Thu and M. Ghanbari. “The accuracy of PSNR in predicting video quality for different video scenes and frame rates”. In: *Telecommun. Syst.* (49 2012), pp. 35–48. DOI: <https://doi.org/10.1007/s11235-010-9351-x>.
- [6] Daniel Jiwoong Im et al. “Denoising Criterion for Variational Auto-Encoding Framework”. In: *CoRR* abs/1511.06406 (2015). arXiv: 1511.06406. URL: <http://arxiv.org/abs/1511.06406>.
- [7] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. URL: <http://arxiv.org/abs/1603.08155>.
- [8] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. “Accurate Image Super-Resolution Using Very Deep Convolutional Networks”. In: *CoRR* abs/1511.04587 (2015). arXiv: 1511.04587. URL: <http://arxiv.org/abs/1511.04587>.
- [9] Diederik P. Kingma and Max Welling. “An Introduction to Variational Autoencoders”. In: *CoRR* abs/1906.02691 (2019). arXiv: 1906.02691. URL: <http://arxiv.org/abs/1906.02691>.
- [10] Debarati Kundu and Brian L. Evans. “Full-reference visual quality assessment for synthetic images: A subjective study”. In: (2015), pp. 2374–2378. DOI: 10.1109/ICIP.2015.7351227.
- [11] Richard McPherson, Reza Shokri, and Vitaly Shmatikov. “Defeating Image Obfuscation with Deep Learning”. In: *CoRR* abs/1609.00408 (2016). arXiv: 1609.00408. URL: <http://arxiv.org/abs/1609.00408>.
- [12] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. “Deep Face Recognition”. In: (Sept. 2015). Ed. by Mark W. Jones Xianghua Xie and Gary K. L. Tam, pp. 41.1–41.12. DOI: 10.5244/C.29.41. URL: <https://dx.doi.org/10.5244/C.29.41>.
- [13] Joseph Rocca. In: (2019). URL: <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>.
- [14] H.R. Sheikh, M.F. Sabir, and A.C. Bovik. “A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms”. In: *IEEE Transactions on Image Processing* 15.11 (2006), pp. 3440–3451. DOI: 10.1109/TIP.2006.881959.
- [15] Jacob Conrad Trinidad. “Reconstructing Obfuscated Human Faces”. In: (). URL: <http://cs231n.stanford.edu/reports/2017/pdfs/223.pdf>.
- [16] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.
- [17] Erjin Zhou, Zhimin Cao, and Qi Yin. “Naive-Deep Face Recognition: Touching the Limit of LFW Benchmark or Not?” In: *CoRR* abs/1501.04690 (2015). arXiv: 1501.04690. URL: <http://arxiv.org/abs/1501.04690>.

7. Appendix

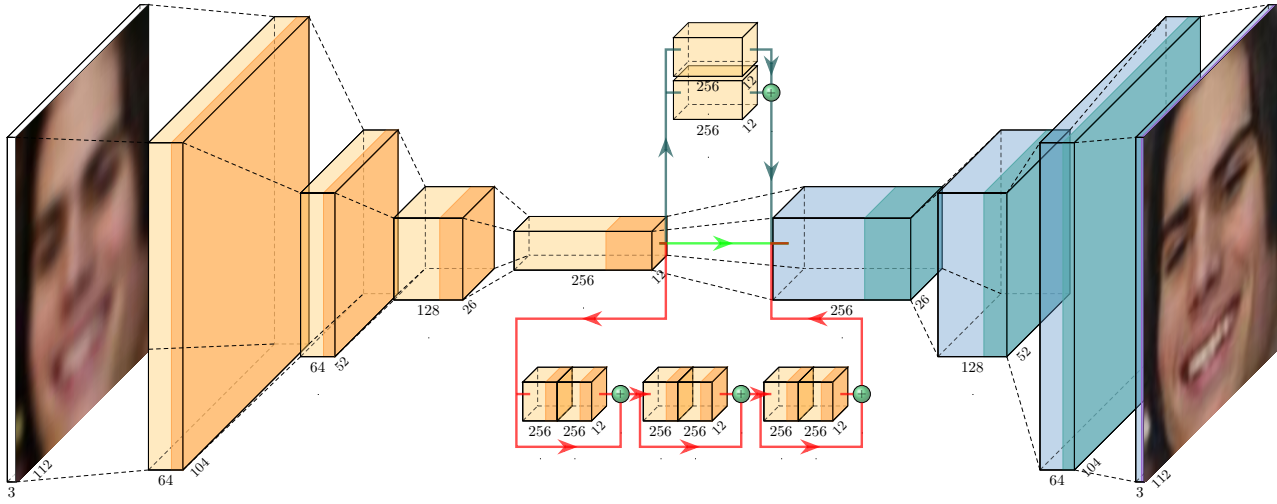


Figure 5: The CAE architecture: the light orange color indicates a Convolutional Layer followed by batch normalization, while the dark orange color refers to ReLU activation applied on a Convolutional Layer. The light blue color indicates a Transposed Convolutional Layer followed by batch normalization; the dark blue color refers to ReLU activation applied on a Transpose Convolutional Layer. The purple color refers to a Tanh activation. The green line indicates the flow of data in the standard architecture, the blue line the one in the variational version, and finally the red one indicates instead the signal path in the architecture deepened with ResBlocks.




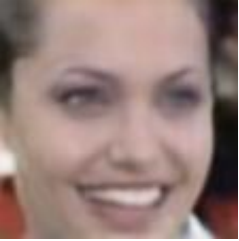




epoch 1	epoch 5	epoch 10	epoch 15
			
24.04 / 0.62	26.82 / 0.76	26.51 / 0.75	27.49 / 0.78
23.19 / 0.62	28.76 / 0.83	29.72 / 0.85	31.07 / 0.87
			
11.52 / 0.21	20.07 / 0.50	24.67 / 0.65	26.55 / 0.72
10.47 / 0.00	18.27 / 0.45	21.94 / 0.64	24.28 / 31.41

Figure 6: Output comparison of a the same image as in 1 across different epochs for the CAE architecture with the pixel loss (first row) and the pure perceptual loss (second row). The PSNR / SSIM for the given example (above) and for the test dataset (below) are reported. Notice how with the perceptual loss the network produces a solid output almost immediately, whose only flaw is the excessive whiteness; this gradually improves as time goes on. Still the perceptual loss produces superb results; for example the teeth separation is already present at epoch 1 and almost perfect at epoch 5 with the perc. loss, even though the pixel loss cannot recover this detail even after 15 epochs.

Once again notice the poor correlation between PSNR / SSIM and human perception of quality.